MAT 115
Exercise #9

In this exercise we are going to work on keeping code and data organized (chs. 39, 40, and 5 from the text).

40.1 Whenever starting something new in RStudio, instead of just creating a new script or markdown file, I prefer using a `Project`. A project is a way to keep all components of a data analysis project (data, scripts, markdown files, README files, etc.) in one folder. If you setup a project, by default any files you create or work on will be saved to that project folder.

For example, on my computer, I have a MAT115 folder that includes separate projects for your homework assignments, in-class exercises, etc.

Your textbook describes the procedure for starting a project in section 40.1. *Try setting up your own MAT115 project.*

If I were you I would move all of my MAT115 associated files into that project.

39 If I know that I am going to be working on a project that will be large, have many versions, involve collaborates, or I will want to share with potential employers, then I will set it up first as a GitHub repository.

Git is a version control system that you run on your own machine. GitHub is a Git repository hosting service. It's been said that GitHub is to Git what Facebook is to your actual face. In short, GitHub does the version control stuff of Git but also allows you to share projects with collaborators or employers.

So, let's start this process by having you *create a Github account*: here. It is a pretty straightforward process but is also described in your book: section 39.2.

The next step is linking your computer to your Github account. Your text recommends doing this by downloading Git and connecting directly through RStudio. However, I like using GitHub Desktop (download here).

You will need to authenticate to GitHub on GitHub Desktop (instructions here).

GitHub Desktop allows you to create Git repositories and link them with GitHub, and it allows you to do so with any IDE or programming language, not just RStudio. I like GitHub Desktop because it has broader use.

We will go through the steps of producing a repository that will ALSO be a folder for an RStudio project, and the process of "cloning" repositories, and "pushing" and "pulling" code updates between RStudio and GitHub. Your text goes into detail about this process in ch. 39. I highly suggest you read through the procedure.

Here is a (link)[https://docs.github.com/en/desktop] with more detail on GitHub Desktop.

Your goal at the end of this section will be to have 1) a GitHub account, 2) a repository/R project with a script or RMarkdown file in it, and 3) practiced pushing and pulling updated repos.

Take a screenshot of the repo you produced. Make sure it includes a record that it has been updated. Turn that screenshot in on Canvas with your knit pdf for this exercise.

Note, GitHub could be very useful if you choose to work with a partner on your midterm project!

5.1 You need to know where your datafiles are, and you need to know how to get to that location using RStudio. If you organize your RStudio usage into projects, like previously described, this should be quite straightforward. Unfortunately, this is not always the case and different computer systems have different ways to indicate a path, so you will have to know the system you are using.

The path of a file is a list of directory names that can be thought of as instructions on what folders to click on, and in what order, to find the file. If these instructions are for finding the file from the root directory

we refer to it as the full path. If the instructions are for finding the file starting in the working directory we refer to it as a relative path.

```
#path to dslabs package:
system.file(package = "dslabs")
```

```
## [1] "C:/Users/Damien Junxi Chiem/AppData/Local/R/win-library/4.4/dslabs"
```

This path is going to be unique to your computer. So, the book recommends that you use relative paths because that makes the code more portable.

For example, this is how I would use a full path to upload the `exams.rda` dataset into RStudio on my computer. Note, if you try and run the code on your computer, you will get an error saying "No such file or directory".

```
#load("C:/Users/ahoward1/Documents/MAT 115/Exercises/exams.rda")
```

Instead, if I use the relative path to my working directory, and assuming the dataset is also in your working directory, the below code should work on both of our machines.

```
dir <- getwd()
print(dir)
```

```
## [1] "C:/Users/Damien Junxi Chiem/OneDrive/Desktop/DSC1/MAT115/M115E9"
```

```
load(file.path(dir,"exams.rda"))
```

You can change the working directory with the function 'setwd()'.

Move your 'exams.rda' dataset to some other location (drag and drop if need be) and try to load it to your session. Use relative paths if at all possible.

```
# Write your code here.
# Remember to use quotation marks "_"

setwd("C:/Users/Damien Junxi Chiem/OneDrive/Desktop/DSC1/MAT115")
load(file.path(dir, "exams.rda"))
```

5.2 It's not likely that your datafile is already in R dataframe format. You will have to convert it to a form that R can work with. We will first assume that the datafile is a simple text file, not in a proprietary format like Microsoft Excel. In this case, there are several tools you can use: base R has `read.table`, `read.csv`, `read.delim` and so on. We will use the `readr` package, which is part of `tidyverse`. See section 5.2 for a list of functions that we can use. You will use the `read_csv` often, since comma-separated datafiles are pretty common.

In order to use `read_csv()` we need to load the `readr` package. The larger `tidyverse` package includes `readr`, so we can load that instead.

```
library(tidyverse)
```

The first thing to do is to look at the file. If necessary, you can edit the file; for example, if there are explanatory text that is not part of the data, you should edit it out. I suggest you look at the file using your computer's text editor, but you can also use the `read_lines` function to take a look.

```
# The following function shows the first 10 lines of the file.
# read_lines("path to the file", n_max=10)

#read_lines("C:/Users/Damien Junxi Chiem/OneDrive/Desktop/DSC1/MAT115/M115E9/AllCountries.csv", n_max=1
```

There should be a `AllCountries.csv` *file on Canvas. Download it, put it somewhere, take a look at it, and then use the* `read_csv` *function to load it into R.*

```
# Write your commands here.
# You should give the file a name, like `countries`.

countries <- read_csv("C:/Users/Damien Junxi Chiem/OneDrive/Desktop/DSC1/MAT115/M115E9/AllCountries.csv
```

```
## Rows: 213 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): Country
## dbl (12): LandArea, Population, Energy, Rural, Military, Health, HIV, Intern...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
countries
```

```
## # A tibble: 213 x 13
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Afghanistan     652230   29021099     NA  76        4.4    3.7  NA        1.7
##  2 Albania          27400    3143291   2088  53.3     NA      8.2  NA       23.9
##  3 Algeria        2381740   34373426  37069  34.8     13     10.6   0.1     10.2
##  4 American Sam~      200      66107     NA   7.7     NA     NA    NA       NA
##  5 Andorra            470      83810     NA  11.1     NA     21.3  NA       70.5
##  6 Angola         1246700   18020668  10972  43.3     NA      6.8   2        3.1
##  7 Antigua and ~      440      86634     NA  69.5     NA     11    NA       75
##  8 Argentina      2736690   39882980  76359   8       NA     13.7   0.5     28.1
##  9 Armenia          28480    3077087   2997  36.1     16.1    7.2   0.1      6.2
## 10 Aruba              180     105455     NA  53.2     NA     NA    NA       22.8
## # i 203 more rows
## # i 4 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>
```

Find the five most populous countries.

```
# Write your code here.
# Remember `arrange` and maybe `select` also.

#countries %>% top_n(5, Population)

countries_arranged <- countries %>% arrange(desc(Population))
countries_arranged[1:5,]
```

```
## # A tibble: 5 x 13
##   Country       LandArea Population  Energy Rural Military Health   HIV Internet
##   <chr>            <dbl>      <dbl>   <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
## 1 China          9327480 1324655000 2116427  56.9     16.1   10.3  NA       22.5
## 2 India          2973190 1139964932  620973  70.5     14.7    4.4   0.3      4.5
## 3 United States  9147420  304375000 2283722  18.3     18.6   18.7   0.6     75.8
## 4 Indonesia      1811570  227345082  198679  48.5      5.3    6.2   0.2      7.9
## 5 Brazil         8459420  191971506  248528  14.4      5.9    6    NA       37.5
## # i 4 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>
```

Now, put the dataset in descending order by population and life expectancy.

```
# Write you code here.
countries %>% arrange(desc(Population), desc(LifeExpectancy))
```

```
## # A tibble: 213 x 13
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 China          9327480 1324655000 2.12e6  56.9     16.1   10.3  NA       22.5
##  2 India          2973190 1139964932 6.21e5  70.5     14.7    4.4   0.3      4.5
##  3 United States  9147420  304375000 2.28e6  18.3     18.6   18.7   0.6     75.8
##  4 Indonesia      1811570  227345082 1.99e5  48.5      5.3    6.2   0.2      7.9
##  5 Brazil         8459420  191971506 2.49e5  14.4      5.9    6    NA       37.5
##  6 Pakistan        770880  166111487 8.28e4  63.8     18      3.1   0.1     11.1
##  7 Bangladesh      130170  160000128 2.79e4  72.9     10.8    7.4   0.1      0.3
##  8 Nigeria         910770  151212254 1.11e5  51.6     10.8    6.4   3.6     15.9
##  9 Russian Fede~ 16376870  141950000 6.87e5  27.2     16.3    9.2   1       32
## 10 Japan           364500  127704000 4.96e5  33.5     NA     17.9   0.1     75.2
## # i 203 more rows
## # i 4 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>
```

If you look carefully at the original file, you should note that there are empty cells, indicating missing data. What happens to these empty cells after you import the data into R?

A: They become NA

Beside plain text files, it is also very common to encounter data in the form of Microsoft Excel files. They are so common that there are packages built to read Excel files directly. (Excel can save files to `.csv` format, so even without these packages, we can still load Excel files into R.) Be warned: Excel files can contain complicated functions and formatting, so the import into R can be problematic.

We will use the `readxl` package, so we first load it.

```
library(readxl)
```

The book lists several functions to read Excel files. The most recent is `read_xlsx` which can be used to read the most recent type of Excel format.

There should also be an Excel file called `AllCountries` on Canvas. Download that, put it somewhere on your computer, and then load it to your workspace using one of the read excel functions. Compare it with what you get when you load the `.csv` file. Are they the same?

```
# Write your code here.

#help(readxl)

countries2 <- read_excel("C:/Users/Damien Junxi Chiem/OneDrive/Desktop/DSC1/MAT115/M115E9/AllCountries.
countries2
```

```
## # A tibble: 213 x 13
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Afghanistan     652230   29021099     NA  76        4.4    3.7  NA        1.7
##  2 Albania          27400    3143291   2088  53.3     NA      8.2  NA       23.9
##  3 Algeria        2381740   34373426  37069  34.8     13     10.6   0.1     10.2
##  4 American Sam~      200      66107     NA   7.7     NA     NA    NA       NA
##  5 Andorra            470      83810     NA  11.1     NA     21.3  NA       70.5
##  6 Angola         1246700   18020668  10972  43.3     NA      6.8   2        3.1
##  7 Antigua and ~      440      86634     NA  69.5     NA     11    NA       75
##  8 Argentina      2736690   39882980  76359   8       NA     13.7   0.5     28.1
##  9 Armenia          28480    3077087   2997  36.1     16.1    7.2   0.1      6.2
## 10 Aruba              180     105455     NA  53.2     NA     NA    NA       22.8
## # i 203 more rows
## # i 4 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>
```

```
countries
```

```
## # A tibble: 213 x 13
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Afghanistan     652230   29021099     NA  76        4.4    3.7  NA        1.7
##  2 Albania          27400    3143291   2088  53.3     NA      8.2  NA       23.9
##  3 Algeria        2381740   34373426  37069  34.8     13     10.6   0.1     10.2
##  4 American Sam~      200      66107     NA   7.7     NA     NA    NA       NA
##  5 Andorra            470      83810     NA  11.1     NA     21.3  NA       70.5
##  6 Angola         1246700   18020668  10972  43.3     NA      6.8   2        3.1
##  7 Antigua and ~      440      86634     NA  69.5     NA     11    NA       75
##  8 Argentina      2736690   39882980  76359   8       NA     13.7   0.5     28.1
##  9 Armenia          28480    3077087   2997  36.1     16.1    7.2   0.1      6.2
## 10 Aruba              180     105455     NA  53.2     NA     NA    NA       22.8
## # i 203 more rows
## # i 4 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>
```

A: The dataframes themselves look the same however, the csv file created a spec_tbl_df rather than a regular tibble. The spec_tbl_df stores more information about the data such as the types of its columns and the specifications.

We should be able to export R data to a plain text format so that others who don't use R can still have access to the data. (Even plain text can have complications: there are ASCII, UTF-8, UTF-16, and other text encodings.) The function to do this is `write_csv()` (or `write_delim`, or `write_tsv`, etc). The basic usage is `write_csv(x,path)` where `x` is the name of an R dataframe (or tibble) and `path` is your intended location of the exported file, including its name. The default path is your working directory.

Try out the ***write_csv*** *function on an R dataset, for example, on the* ***exams*** *dataframe. Don't forget to* ***load*** *the exams dataset into your workspace first. Check that the function works.*

```
# Write your code here.
write_csv(exams, "exams.txt")
```

Here are some more exercises about manipulating dataframes. You should have the ***AllCountries*** *dataframe in your R session by now. Try doing the following things (these are separate tasks):*

- add a new column ***density*** *which shows the number of people per sq kilometer. Then find the top 10 densest countries. Also the 10 least dense countries.*

```
countries_mod <- mutate(countries, density=(Population/LandArea))

#countries_mod$density <- countries$Population / countries$LandArea

countries_mod
```

```
## # A tibble: 213 x 14
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Afghanistan     652230   29021099     NA  76        4.4    3.7  NA        1.7
##  2 Albania          27400    3143291   2088  53.3     NA      8.2  NA       23.9
##  3 Algeria        2381740   34373426  37069  34.8     13     10.6   0.1     10.2
##  4 American Sam~      200      66107     NA   7.7     NA     NA    NA       NA
##  5 Andorra            470      83810     NA  11.1     NA     21.3  NA       70.5
##  6 Angola         1246700   18020668  10972  43.3     NA      6.8   2        3.1
##  7 Antigua and ~      440      86634     NA  69.5     NA     11    NA       75
##  8 Argentina      2736690   39882980  76359   8       NA     13.7   0.5     28.1
##  9 Armenia          28480    3077087   2997  36.1     16.1    7.2   0.1      6.2
## 10 Aruba              180     105455     NA  53.2     NA     NA    NA       22.8
## # i 203 more rows
## # i 5 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>, density <dbl>
```

- construct a new dataframe that consists only of health-related data. (Of course, you should include the name of the countries too.)

```
countries_health <- select(countries, Country, Health, HIV, BirthRate, LifeExpectancy)

countries_health
```

```
## # A tibble: 213 x 5
##    Country             Health   HIV BirthRate LifeExpectancy
##    <chr>                <dbl> <dbl>     <dbl>          <dbl>
##  1 Afghanistan            3.7  NA        46.5           43.9
##  2 Albania                8.2  NA        14.6           76.6
##  3 Algeria               10.6   0.1      20.8           72.4
##  4 American Samoa        NA    NA        NA             NA
##  5 Andorra               21.3  NA        10.4           NA
##  6 Angola                 6.8   2        42.9           47
##  7 Antigua and Barbuda   11    NA        NA             NA
```

```
##  8 Argentina             13.7   0.5      17.3           75.3
##  9 Armenia                7.2   0.1      15.3           73.5
## 10 Aruba                 NA    NA        11.7           74.7
## # i 203 more rows
```

- some of the data are missing. For example, the variable **`Developed`** *classifies countries into categories 1, 2, or 3. (What are they?) But not all countries are included. Dominica, for example, does not have a classification. Construct a dataframe that includes only countries that are classified and also where we know the percentage of the elderly population.*

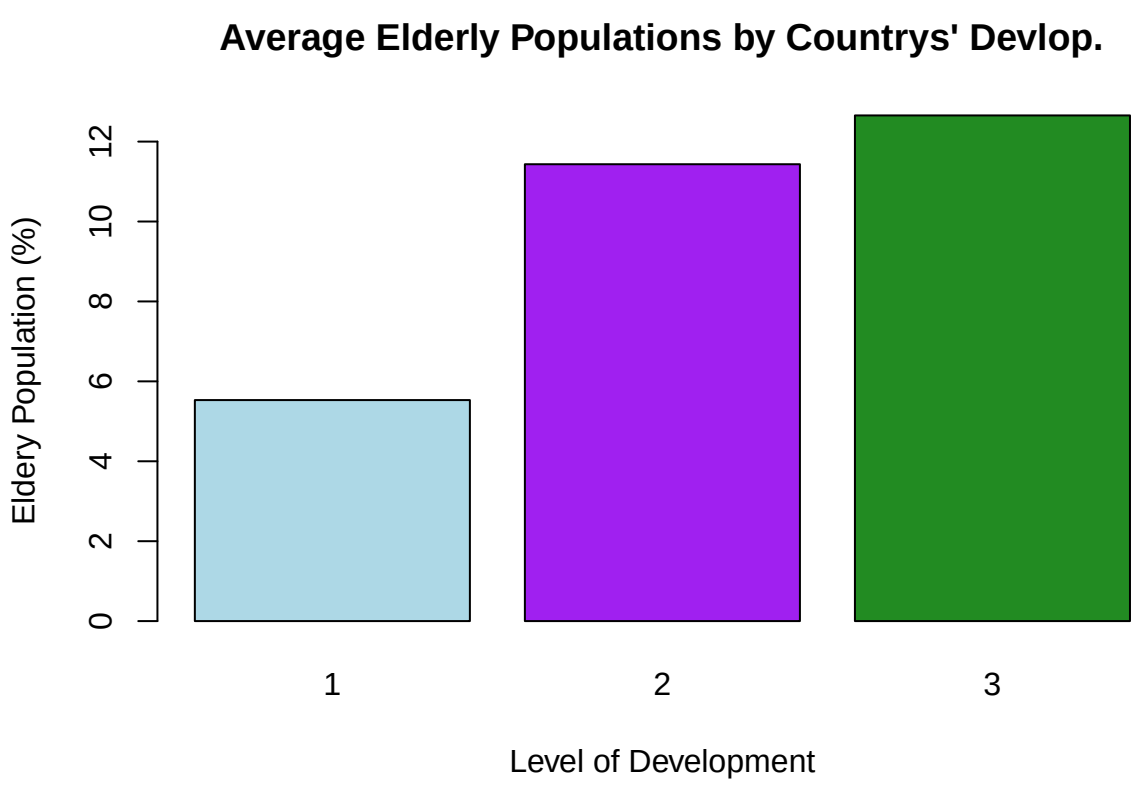
A: 1-3, 1 as not developed, 2 as developing, and 3 as developed

```
countries_classified <- filter(countries, !is.na(Developed) & !is.na(ElderlyPop))

countries_classified
```

```
## # A tibble: 134 x 13
##    Country    LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>         <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Albania       27400    3143291   2088  53.3     NA      8.2  NA       23.9
##  2 Algeria     2381740   34373426  37069  34.8     13     10.6   0.1     10.2
##  3 Angola      1246700   18020668  10972  43.3     NA      6.8   2        3.1
##  4 Argentina   2736690   39882980  76359   8       NA     13.7   0.5     28.1
##  5 Armenia       28480    3077087   2997  36.1     16.1    7.2   0.1      6.2
##  6 Australia   7682300   21431800 130113  11.3      7.7   17.1   0.1     70.8
##  7 Austria       82450    8336926  33246  32.8      2.4   15.8   0.3     72.9
##  8 Azerbaijan    82620    8680100  13367  48.1     22.9    2.5   0.1     28.2
##  9 Bahrain         760     775585   9226  11.5     15.6   10.3  NA       51.9
## 10 Bangladesh   130170  160000128  27944  72.9     10.8    7.4   0.1      0.3
## # i 124 more rows
## # i 4 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>
```

- continuation of the previous exercise. Compare the average percent of elderly population among the three different types of countries. Produce a plot to make the comparison visually.

```
grouped_by_dev <- countries_classified %>%
  group_by(Developed) %>%
  summarize(AvgElderly = mean(ElderlyPop))

barplot(grouped_by_dev$AvgElderly,
        names.arg = grouped_by_dev$Developed,
        col=c("lightblue", "purple", "forestgreen"),
        main = "Average Elderly Populations by Countrys' Devlop.",
        xlab = "Level of Development",
        ylab = "Eldery Population (%)")
```



- add a new column which essentially replaces `1` in the `Developed` column by "least developed",`2` by "developing', and `3` by"developed". Complete this task using both tidyverse and base R syntax.

```
#tidy verse method
tidy_countries <- countries %>% mutate(Alpha_Developed = case_when(
  Developed == 1 ~ "least developed",
  Developed == 2 ~ "developing",
  Developed == 3 ~ "developed",
  TRUE ~ NA
))

tidy_countries
```

```
## # A tibble: 213 x 14
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Afghanistan     652230   29021099     NA  76        4.4    3.7  NA        1.7
##  2 Albania          27400    3143291   2088  53.3      NA      8.2  NA       23.9
##  3 Algeria        2381740   34373426  37069  34.8      13     10.6   0.1     10.2
##  4 American Sam~      200      66107     NA   7.7      NA     NA    NA       NA
##  5 Andorra            470      83810     NA  11.1      NA     21.3  NA       70.5
##  6 Angola         1246700   18020668  10972  43.3      NA      6.8   2        3.1
##  7 Antigua and ~      440      86634     NA  69.5      NA     11    NA       75
##  8 Argentina      2736690   39882980  76359   8        NA     13.7   0.5     28.1
##  9 Armenia          28480    3077087   2997  36.1      16.1    7.2   0.1      6.2
## 10 Aruba              180     105455     NA  53.2      NA     NA    NA       22.8
```

```
## # i 203 more rows
## # i 5 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>, Alpha_Developed <chr>
```

```
#base r method

R_countries <- countries

development <- vector(length = length(countries$Developed))

for(i in 1:length(development)){

  if(is.na(countries$Developed[i])){
    development[i] = NA
  } else {
    if(countries$Developed[i] == 1){
      development[i] = "least developed"
    } else if (countries$Developed[i] == 2) {
      development[i] = "developing"
    } else {
      development[i] = "developed"
    }
  }
}

R_countries$Alpha_Development <- development

R_countries
```

```
## # A tibble: 213 x 14
##    Country       LandArea Population Energy Rural Military Health   HIV Internet
##    <chr>            <dbl>      <dbl>  <dbl> <dbl>    <dbl>  <dbl> <dbl>    <dbl>
##  1 Afghanistan     652230   29021099     NA  76        4.4    3.7  NA        1.7
##  2 Albania          27400    3143291   2088  53.3     NA      8.2  NA       23.9
##  3 Algeria        2381740   34373426  37069  34.8     13     10.6   0.1     10.2
##  4 American Sam~      200      66107     NA   7.7     NA     NA    NA       NA
##  5 Andorra            470      83810     NA  11.1     NA     21.3  NA       70.5
##  6 Angola         1246700   18020668  10972  43.3     NA      6.8   2        3.1
##  7 Antigua and ~      440      86634     NA  69.5     NA     11    NA       75
##  8 Argentina      2736690   39882980  76359   8       NA     13.7   0.5     28.1
##  9 Armenia          28480    3077087   2997  36.1     16.1    7.2   0.1      6.2
## 10 Aruba              180     105455     NA  53.2     NA     NA    NA       22.8
## # i 203 more rows
## # i 5 more variables: Developed <dbl>, BirthRate <dbl>, ElderlyPop <dbl>,
## #   LifeExpectancy <dbl>, Alpha_Development <chr>
```